Students:
**Nguyen Huu Truong Giang** [u6110634@udg.edu]
**Praveen Kumar Murali** [u6110748@udg.edu]

# Lab 2: Potential Functions

## 1 Introduction

Potential fields is a path planning algorithm that models the robot and obstacles as positively entities, while the goal position is assigned a negative charge. As a result, the robot is attracted toward the goal and rebelled by the obstacles, forming a trajectory as shown in Figure 1.

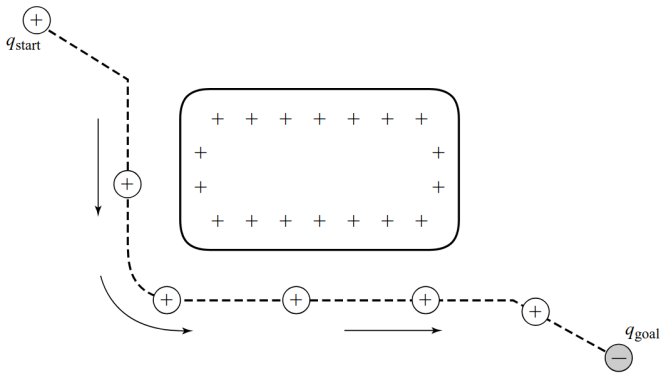


Figure 1: The path is formed because the robot is attracted to the negative charge and repelled by the positive charge [1]

To implement the potential field method, we first compute the attractive potential associated with the goal and the repulsive potential generated by the obstacles. The total potential map is then obtained by summing these two components as shown in Figure 2.

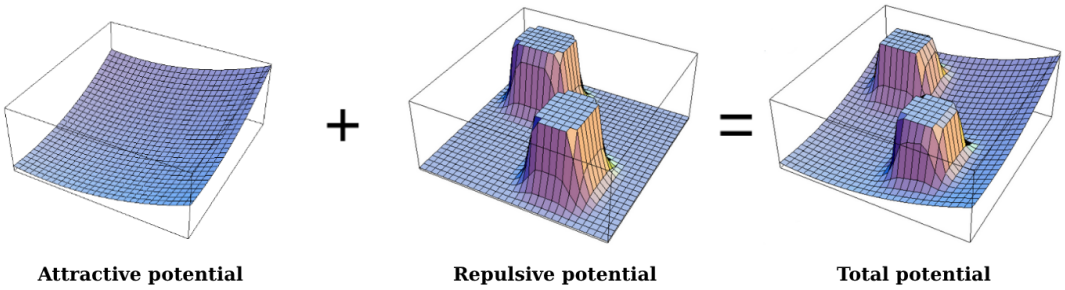


Figure 2: The summing of attractive potential and repulsive potential to produce the total potential [2]

This report is organized as follows: Section 2 presents the mathematical formulation of the algorithm and the pseudocode used to implement it. Section 3 discusses the results obtained from different maps and algorithms to evaluate the performance and the limitations of each method in different scenarios. Finally, Section 4 is devoted to the description of the problems founds during the lab, the conclusion about the local minimums obtained from the results and how to avoid it.

# 2 Methodology

This section presents the mathematical equation and pseudocode of potential field method, including both the attractive and repulsive functions, as well as the gradient descent algorithm used to find a path from start to goal position on a given potential field map with different connectivity strategies (4-connectivity and 8-connectivity). Moreover, the wave-front planner method is introduced as an alternative algorithm to ensure a path to the goal can be found without encountering the local minima problem.

## 2.1 Attractive potential field

The quadratic attractive potential energy is defined as:

$$U_{att}(q) = \frac{1}{2}\zeta d^2(q, q_{goal}) \tag{1}$$

Where:

- $q$: robot configuration.
- $q_{goal}$: goal configuration.
- $d(q, q_{goal})$: Euclidean distance between $q$ and $q_{goal}$.
- $\zeta$: positive scaling factor controlling the strength of attraction.

To compute the attractive potential map, we first calculate the Euclidean distance from each point in configuration space to the goal position. Then, using the Equation (1), the attractive potential energy $U_{att}$ for each point in configuration space is determined. The pseudocode for implementing this process is shown in Algorithm 1.

**Algorithm 1** Computation of the attractive potential field

Create a 2D attractive potential field map $U_{att}$ with the same size as the environment
Initialize the positive scaling factor $\zeta$
**for** each cell q in the grid map **do**
    $d(q, q_{goal}) \leftarrow EuclideanDistance(q, q_{goal})$
    $U_{att}[q] \leftarrow \frac{1}{2}\zeta d^2(q, q_{goal})$
**end for**
**return** $U_{att}$

## 2.2 Repulsive potential field

The repulsive potential keeps the robots away from the obstacles and its' equation is defined below:

$$U_{req}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{D(q)} - \frac{1}{Q^*}) & , D(q) \leq Q^* \\ 0 & , D(q) > Q^* \end{cases} \tag{2}$$

Where:

- $D(q)$: Distance between the robot and the closest obstacle.
- $Q^*$: Threshold distance beyond which the repulsive potential is zero.
- $\eta$: Positive constant controlling the strength of repulsion.

To obtain the repulsive potential map, we first calculate the distance from each point in the map to the nearest obstacle using the **Brushfire** algorithm. Before performing this computation, two types of connectivity must be defined: **4-connectivity** and **8-connectivity**, as illustrated in Figure 3.

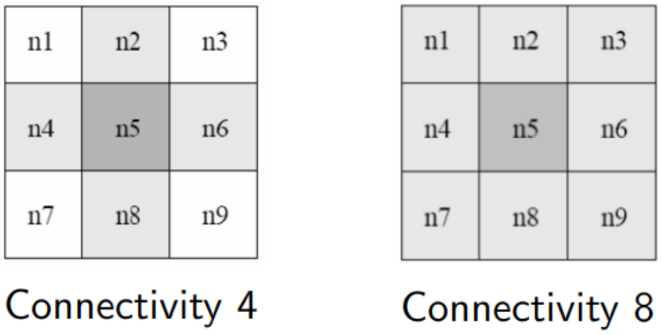


Figure 3: Representation of 4 and 8-connectivity

To implement the **Brushfire** algorithm, we first initialize the distance map $DM_{obs}$ where obstacle cells are assigned a value of 1 and free-space cells are assigned 0. Next, we create a queue $L$ that stores all boundary cells adjacent to obstacles. Each cell in the queue is then processed iteratively. For each extracted cell, we check its neighboring cells based on the selected connectivity type (4 or 8). If a neighboring cell has a value of 0, it is assigned the value of the current cell plus 1 and added to the queue. This process continues until the queue is empty. As a result, we obtain the distance map to the nearest obstacle $DM_{obs}$. The pseudocode for implementing the **Brushfire** algorithm is provided in Algorithm 2.

After getting the distance map to the nearest obstacle $DM_{obs}$ using **Brushfire** algorithm, we define value $Q^*$ and $\eta$ to use the equation (2) for computing the repulsive potential value for each cell of the map. The pseudocode for implementing repulsive potential function is shown in Algorithm 3.

**Algorithm 2** Brushfire algorithm

Initialize distance map $DM_{obs}$: set obstacle cells to 1 and free cells to 0
Create a queue $L$ with all boundary cells adjacent to obstacles
**while** $L$ is not empty **do**
    Pop the front element $t$ from $L$
    **for** each neighboring cell $n$ of $t$ **do**
        **if** $DM_{obs} = 0$ **then**
            $DM_{obs}[n] \leftarrow DM_{obs}[t] + 1$
            Add $n$ to $L$
        **end if**
    **end for**
**end while**
**return** $DM_{obs}$

**Algorithm 3** Computation of the repulsive potential field

**Input**: Distance to nearest obstacle map $DM_{obs}$
Create a 2D repulsive potential field map $U_{req}$ with the same size as the environment
Choose distance threshold $Q^*$, scaling factor $\eta > 0$
**for** each cell $q$ in $DM_{obs}$ **do**
    **if** $DM_{obs}[q] \leq Q^*$ **then**
        $U_{req}(q) \leftarrow \frac{1}{2}\eta(\frac{1}{DM_{obs}[q]} - \frac{1}{Q^*})$
    **else if** $DM_{obs}[q] > Q^*$ **then**
        $U_{req}(q) \leftarrow 0$
    **end if**
**end for**
**return** $U_{req}$

## 2.3 Total potential field and gradient descent

From the attractive and repulsive potential fields in Equations (1) and (2), respectively, the total potential map can be computed by summing the two components, as expressed below:

$$U_{total}(q) = U_{att}(q) + U_{rep}(q) \tag{3}$$

Once the total potential field map is obtained, the robot follows the negative gradient toward the goal. In this lab, the **Discrete Gradient Approximation** method is used to guide the robot from its start position to the goal. This method operates in three main steps:

1. For the current position, compare its potential value with its 4 or 8 neighboring cells, depending on the chosen connectivity.
2. Select the neighboring cell with the smallest potential value.
3. Move the robot to that neighboring cell.

The pseudocode for implementing the **Discrete Gradient Approximation** algorithm is provided in Algorithm 4.

**Algorithm 4** Discrete Gradient Approximation

**Input**: The total potential field map $U_{total}$, start point $q_{start}$, goal point $q_{goal}$
$q \leftarrow q_{start}$, $i \leftarrow 0$
**while** $i < i_{max}$ **do**
    $Min\_Value \leftarrow 0$, $next\_cell \leftarrow q$
    **for** each neighbor $n$ of $q$ **do**
        $\nabla U(n) \leftarrow U(n) - U(q)$
        **if** $\nabla U(n) < Min\_value$ **then**
            $Min\_value \leftarrow \nabla U(n)$
            $next\_cell \leftarrow n$
        **end if**
    **end for**
    $q \leftarrow next\_cell$ //Move to next_cell
**end while**

## 2.4 Wave-front Planner

The **Wave-front Planner** is actually the **Brushfire** algorithm but it apllies from the goal position instead of obstacles. To implement this algorithm, we first initialize the map as follows: obstacle cells are assigned a value of infinity, free-space cells are assigned a large value, and the goal cell is assigned a value of zero. A queue $Q$ is then created to store the goal position. The algorithm proceeds by repeatedly extracting the first element from $Q$ and examining its neighboring cells based on the selected connectivity type (4-connectivity or 8-connectivity). If a neighboring cell's value is not infinity and its current value is greater than the value of the current cell plus one, it is

updated accordingly. This process continues until the queue $Q$ becomes empty. The pseudocode for implementing the **Wave-Front Planner** algorithm is presented in Algorithm 5.

**Algorithm 5** Wave-front Planner

Create a 2D distance map $DM$ with the size of the environment.
Initialize free cells to $MAX\_VALUE$ (unvisited).
Initialize obstacle cells to inf (unreachable).
**while** $Q$ is not empty **do**
    Pop $t$ from $Q$
    **for** each neighbor n of t **do**
        **if** $DM(n)$ is not inf and $DM(n) > DM(t) + 1$ **then**
            Set $DM(n) = DM(t) + 1$
            Add $n$ to $Q$
        **end if**
    **end for**
**end while**
**return** $DM$

Once the distance map $DM$ is obtained, the **Greedy Gradient Descent on Grid** method is used to determine the path from the start to the goal position. This method begins at the start position and iteratively examines all neighboring cells (based on the chosen connectivity). At each step, the robot moves to the neighboring cell with the smallest distance value and continues until it reaches the goal position. The pseudocode for the **Greedy Gradient Descent on Grid** algorithm is shown in Algorithm 6.

**Algorithm 6** Greedy Gradient Descent On Grid

**Input:** distance map $DM$, start cell $q_{start}$ , goal cell $q_{goal}$.
Set $q \leftarrow q_{start}$
**while** $q \neq q_{goal}$ **do**
    Set $best\_value = DM(q)$, $next\_cell = q$.
    **for** each neighbor n of t **do**
        **if** $DM(n) < best\_value$ **then**
            $best\_value = DM(n)$
            $next\_cell = n$
        **end if**
    **end for**
    $q \leftarrow next\_cell$ // Move to next cell
**end while**

# 3 Results

We conducted the experiments using the **Potential Field** and **Wave-front Planner** methods on four maps (Map 0, 1, 2, and 3). In each map, the green circle represents the start position, and the red star indicates the goal position, as shown in Figure 4 and Figure 5. These experiments allow us to perform a general evaluation and comparison between the two path planning methods.

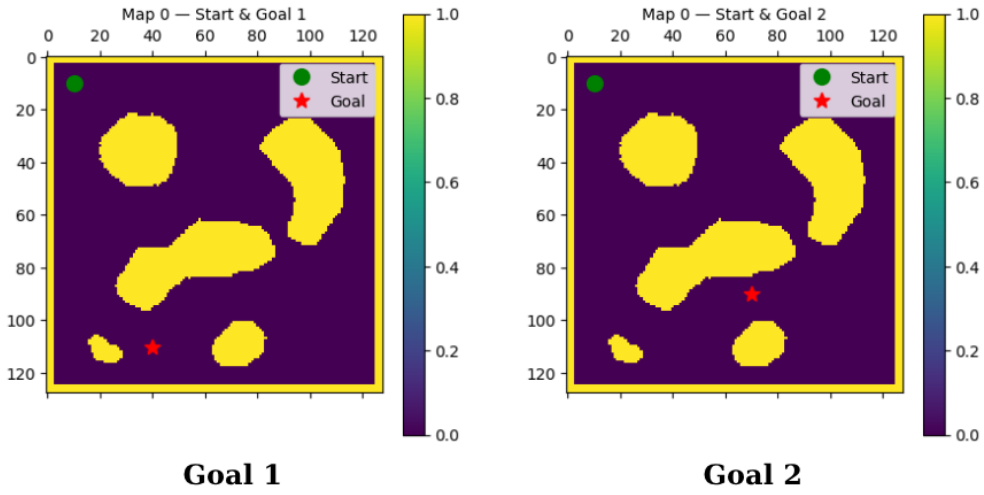


Figure 4: Map 0 with different goal positions

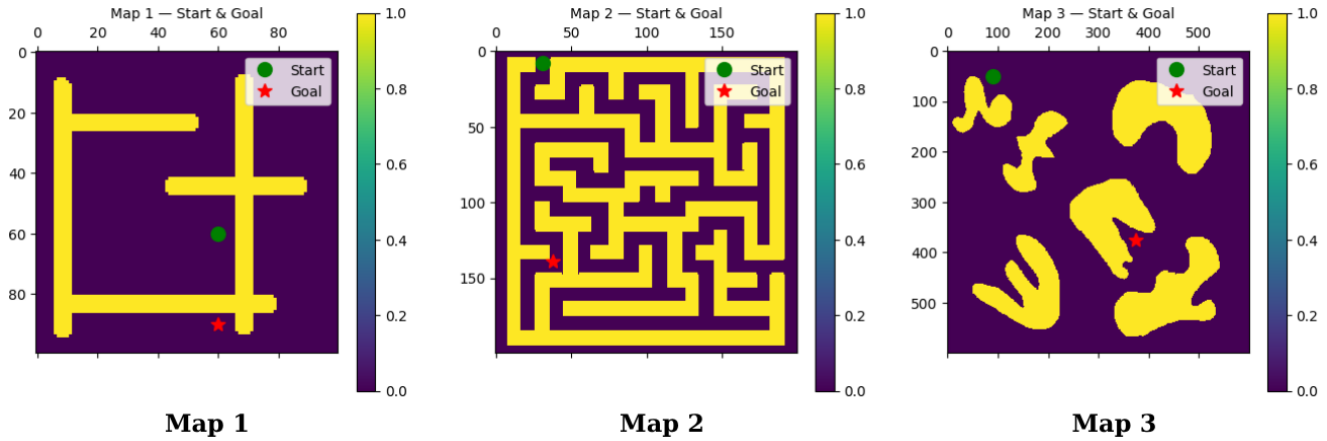


Figure 5: Map 1, 2, and 3

## 3.1 Potential Field

In this lab, we applied the **Potential Field** method to four different maps using two types of connectivity: **4-connectivity** and **8-connectivity**. The parameters used on each map are summarized in Table 1.

| Map | Start | Goal | $\zeta$ | $\eta$ | $Q^*$ | num iterations |
|---|---|---|---|---|---|---|
| 0 | $[10, 10]$ | $[110, 40]$ | 0.01 | 100 | 10 | 1000 |
| 0 | $[10, 10]$ | $[90, 70]$ | 0.01 | 100 | 10 | 1000 |
| 1 | $[60, 60]$ | $[90, 60]$ | 0.01 | 100 | 10 | 1000 |
| 2 | $[8, 31]$ | $[139, 38]$ | 0.01 | 100 | 10 | 1000 |
| 3 | $[50, 90]$ | $[375, 375]$ | 0.01 | 100 | 10 | 1000 |

Table 1: Parameter configuration for each map

### 3.1.1 4-connectivity

From the experimental results shown in Figure 6 and Figure 7, it can be observed that in **Map 0** with the goal position at $[110, 40]$ (Goal 1), the path (**red path**) from the start to the goal position is successfully found. However, when the goal position is changed to $[90, 70]$, the robot fails to reach the goal because it becomes trapped in a local minimum, where the gradient value $\nabla U = 0$. Furthermore, in **Maps 1, 2, and 3**, the robot also becomes stuck in local minima, preventing it from reaching the goal position.

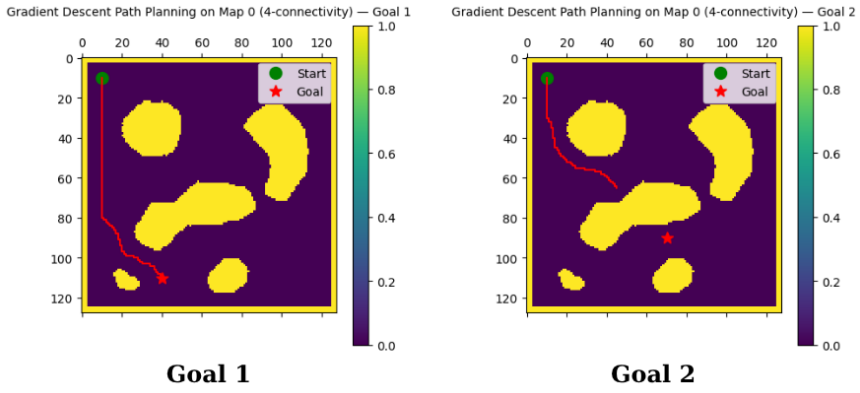


Figure 6: Path finding using potential field (4-connectivity) in map 0 with different goal positions

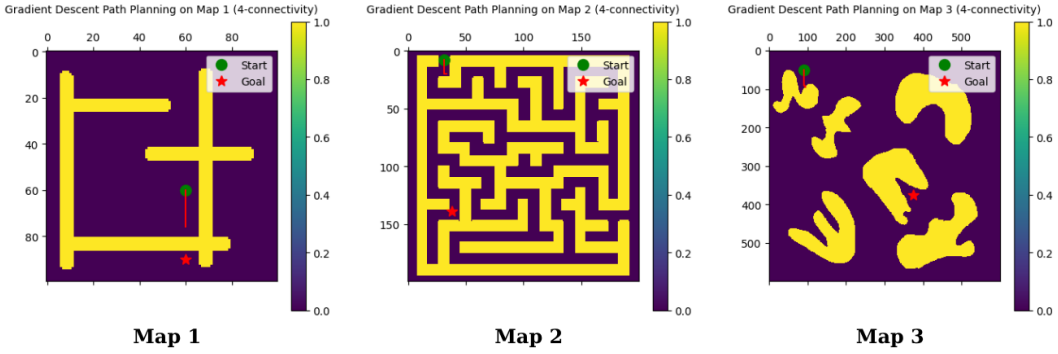


Figure 7: Path finding using potential field (4-connectivity) in map 1, 2, and 3

### 3.1.2 8-connectivity

With 8-connectivity, robot is unable to find the path to the goal position in all four maps since it is stuck at local minimum where the gradient value is 0. The corresponding results are illustrated in Figure 8 and Figure 9.

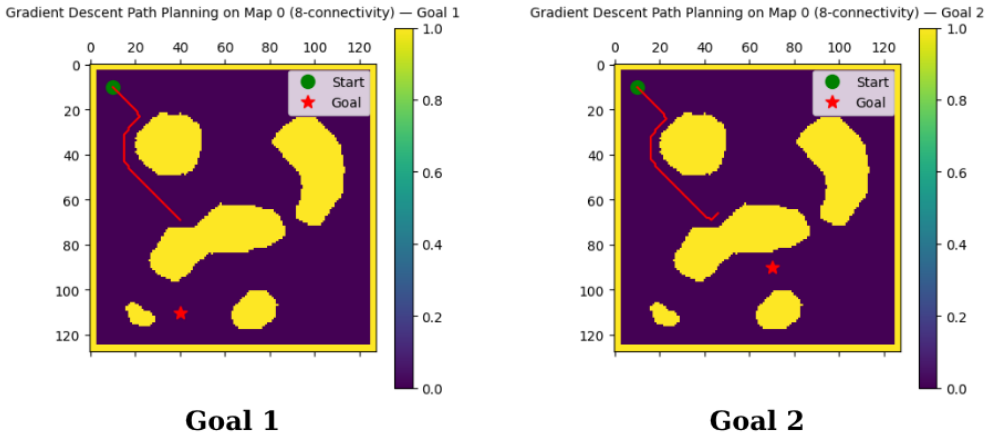


Figure 8: Path finding using potential field (8-connectivity) in map 0 with different goal positions

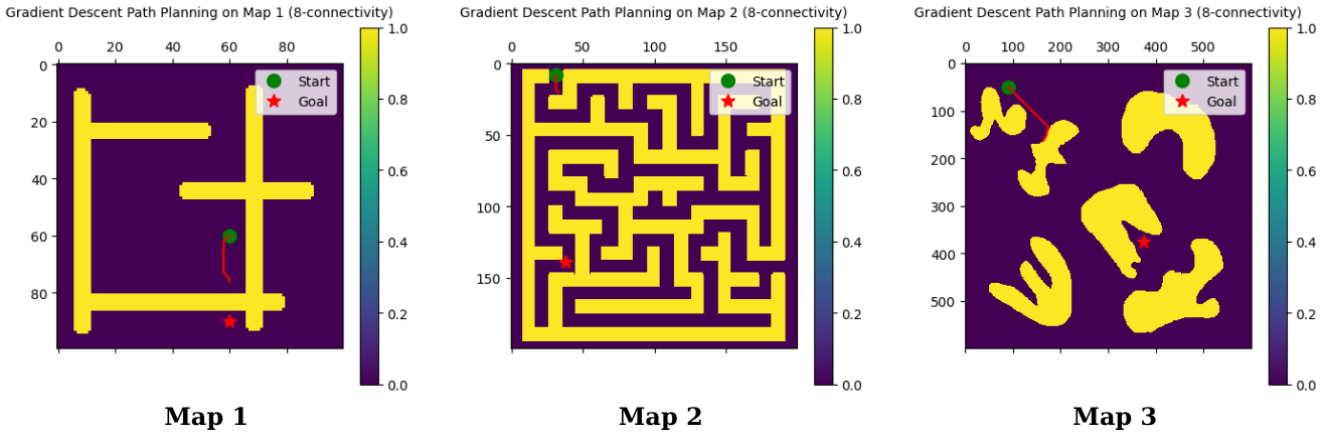


Figure 9: Path finding using potential field (8-connectivity) in map 1, 2, and 3

## 3.2 Wave-front planner

With **Wave-front Planner** method, the approach is similar to the **Brushfire** algorithm. However, it is applied from the goal position instead of obstacles, and the result paths are different based on the type of connectivity (4 or 8-connectivity). The start and goal configuration in each map are described in Table 1.

### 3.2.1 4-connectivity

As you can see in Figure 10 and Figure 11, the robot is able to find the path to the goal in all four maps for 4 connectivity.

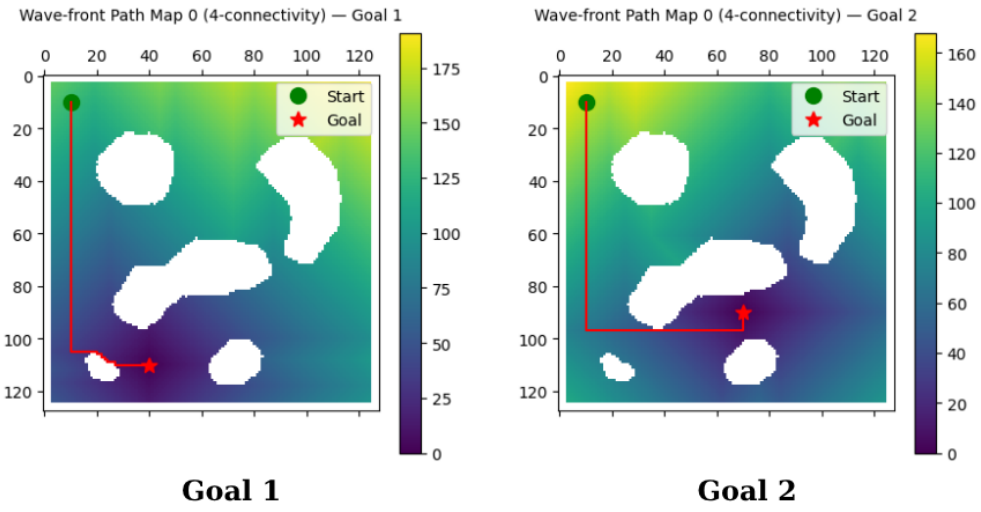


Figure 10: Wave-front planner (4-connectivity) in map 0 with different goal positions

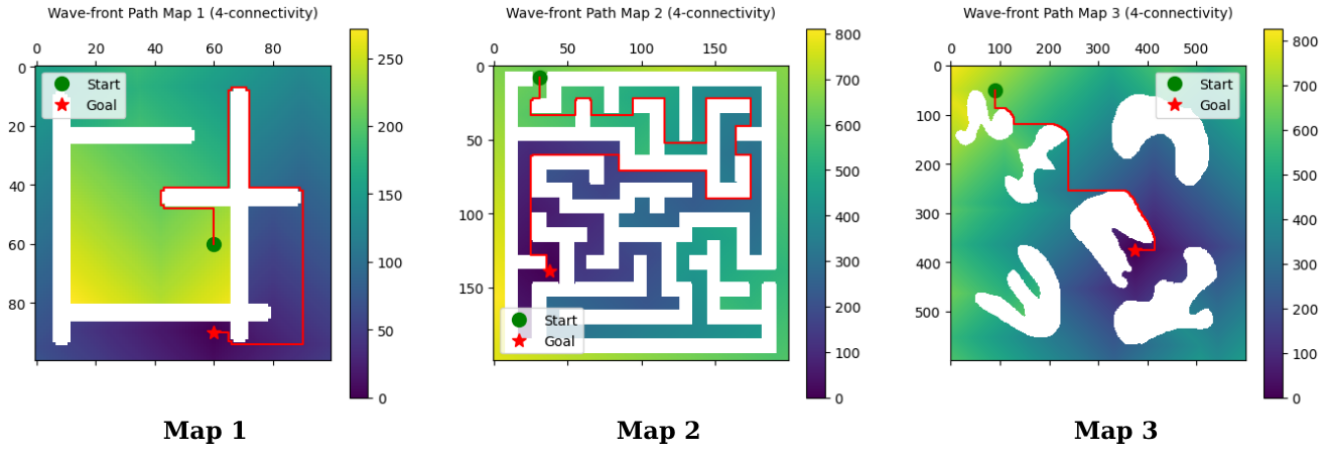


Figure 11: Wave-front planner (4-connectivity) in map 1, 2, and 3

### 3.2.2 8-connectivity

Similar to the results obtained using 4-connectivity, the robot successfully finds a path to the goal position in all four maps but the path is different since the robot now can move in 8 directions instead of 4 directions. The results are illustrated in Figure 12 and Figure 13.

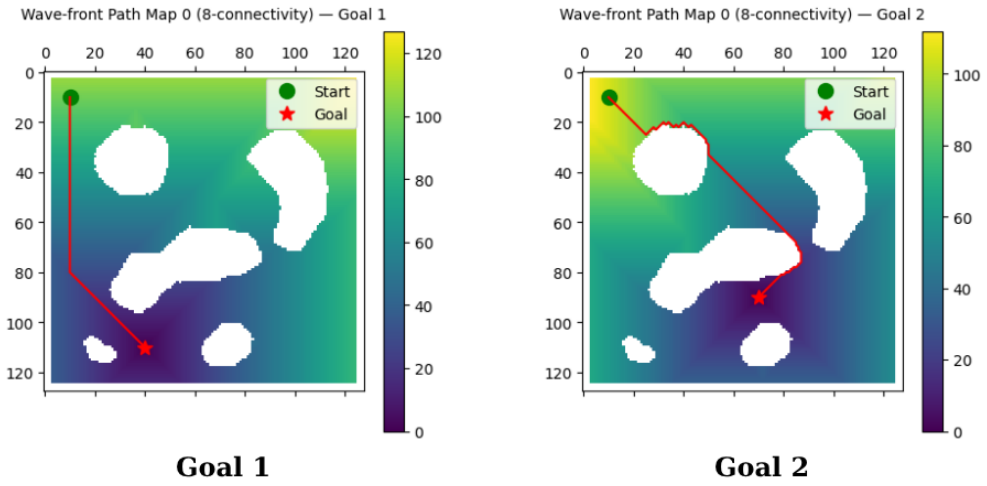


Figure 12: Wave-front planner (8-connectivity) in map 0 with different goal positions

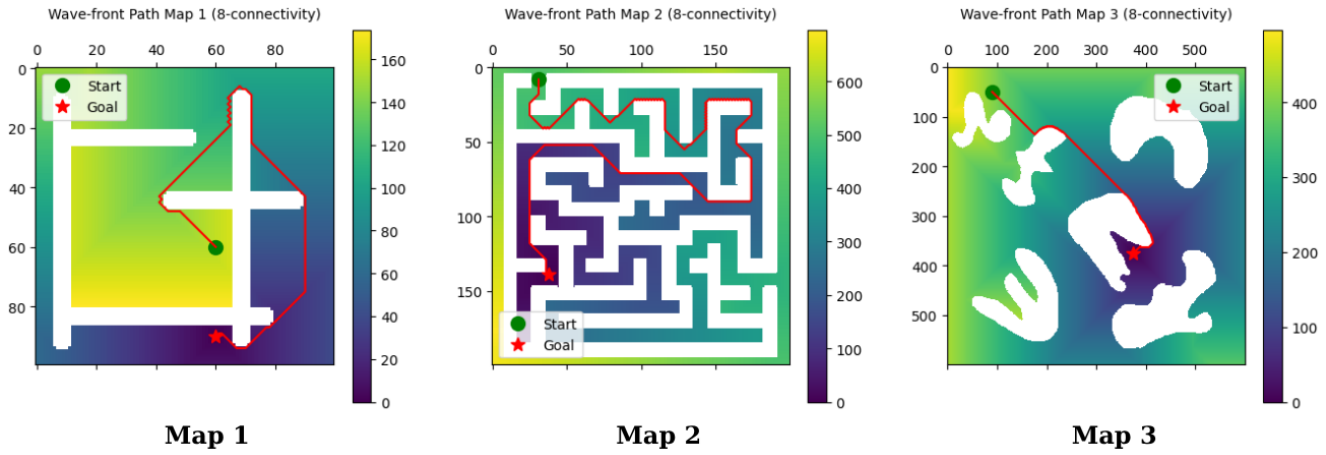


Figure 13: Wave-front planner (8-connectivity) in map 1, 2, and 3

# 4 Discussions and Conclusions

The **Potential Field** method is an intuitively understandable algorithm that models the robot as a positively charged entity attracted to the goal, which is represented as a negatively charged entity. Along its path, the robot is repelled by obstacles that are also assigned positive charges. However, a major drawback of the **Potential Field** method is that it can cause the robot to become trapped in local minima, as the results shown in Section 3.1.

In the case of 4-connectivity, the robot successfully reaches the goal position in Map 0, but when the goal position is changed, the robot becomes trapped in a local minimum. Furthermore, when applied to more complex environments such as Map 1, Map 2, and Map 3, the robot consistently gets stuck in local minima. Similarly, when using 8-connectivity, the **Potential Field**

method fails to reach the goal in any of the maps, including Goal 1 in Map 0. Therefore, the **Potential Field** algorithm is more suitable for simple environments where local minima are not present.

To ensure the robot reliably reaches the goal and avoids local minima, a path planning algorithm that guarantees only a global minimum should be used, such as the **Wave-front Planner**. The **Wave-front Planner** expands a wavefront from the goal position throughout all free cells, ensuring the existence of a single global minimum. As demonstrated in Section 3.2, this method successfully finds a path to the goal position in all four maps, even in complex environments.

In summary, the **Potential Field** method is suitable for simple environments without local minima, whereas the **Wave-front Planner** is more robust in complex environments, ensuring convergence to a global minimum.

# References

[1] N. Palomeras, "Slide autonomous systems potential functions," *Universitat de Girona*. `https://moodle.udg.edu/pluginfile.php/2491455/mod_resource/content/5/potential_functions.pdf`.

[2] H. Choset, "Slide robotic motion planning: Potential functions," `https://www.cs.cmu.edu/~motionplanning/lecture/Chap4-Potential-Field_howie.pdf`.